Trabajo Práctico Final:

Programación Orientada a Objetos

Integrantes

Baliarda Gonzalo 61490,

Pérez Ezequiel Agustín 61475

Introducción

Este informe detalla las modificaciones, implementaciones y dificultades encontradas al momento de realizar los distintos modos de juego y su correcta funcionalidad para una versión académica y reducida del juego “Candy Crush”.

Funcionalidades agregadas

1. Opción en la barra de menú que despliega los distintos modos de juego disponibles.

2. Efecto visual al seleccionar el caramelo con el que se va a ejecutar el movimiento deseado.

3. Alerta al ganar o perder en cualquiera de los modos de juego habilitados.

4. Para el modo de juego “Clásico”, la cantidad de movimientos restantes y el puntaje necesario para “ganar”.

5. Modo de juego “GoldenBoard”, en el que hay que “pintar” el tablero de color dorado mediante movimientos, donde muestra la cantidad de movimientos restantes y las celdas que aún faltan pintar.

6. Modo de juego “TimeBomb”, en el que hay que eliminar los caramelos con contador antes de que lleguen a 0, donde cada movimiento que no elimina a estos caramelos le resta 1 a su contador. La implementación incluye la cantidad de caramelos que faltan eliminar para ganar, junto al puntaje.

7. Modo de juego “TimeLimit”, en el que hay que llegar a un puntaje determinado antes de que el contador de tiempo llegue a 0. En este nivel se cuenta con caramelo con un texto por encima que indica la cantidad de segundos que le agrega al contador.

Dificultades durante el desarrollo

Durante el desarrollo del Trabajo Práctico nos encontramos con cierta complicación al comprender el código provisto por la Cátedra ya que eran muchos archivos, sumado a dónde y cómo realizar las implementaciones en el código dado.

Además, algunas partes del código eran difíciles de escalar hacia nuevas funcionalidades, y debían realizarse casteos que complicaban las cosas.

Cambios en la implementación provista

Frontend

- En la carpeta resources, se agregó “LogoCandy.png” que se utiliza como logo de la aplicación.

- En la clase ‘GameApp’ se implementó la barra de menú (en vez de en CandyFrame como en el original) para que no se cargue la barra de menú cada vez que se cambia de nivel de juego y también arregla un problema que había al traducirse las coordenadas ya que en el eje ‘Y’, el 0 estaba dentro del menú y no en el tablero. Además se le agregó un título, un logo y la escena de cada nivel.

- La clase ‘CandyFrame’ se hizo abstracta. En ella se implementan funciones que utilizan todos los niveles, además de otras que, según el nivel, se sobreescriben en el CandyFrame de cada nivel según sea necesario.

  Parte del código que previamente estaba en el constructor fue modularizado, junto con otros cambios para evitar la repetición de código y el correcto funcionamiento del resto de los niveles. Lo más importante a destacar es la función “endScreen” que, como su nombre indica, muestra la pantalla al ganar o perder en un nivel.

- La clase ‘BoardPanel’, al igual que ‘CandyFrame’ se hizo abstracta para adquirir una mejor modularización y evitar repetir código, y se pasó el código que antes cargaba la imagen en las celdas a las clases hijas de esta donde cada una lo implementa como lo necesita. También se definen los métodos abstractos del efecto visual al seleccionar un caramelo.

- La clase ‘ScorePanel’ se hizo abstracta y recibió un cambio en cómo se muestra el puntaje, con un Label que indica el puntaje y el puntaje necesario para ganar, en caso de que exista tal puntaje.

- En la clase ‘AppMenu’ se añade la sección en la que se muestra la lista de niveles disponibles. También se modificó el tamaño de la letra, y en el menú ‘Acerca De’ se agregó el nombre de los integrantes que realizaron los cambios sobre la implementación original provista.

- Se crearon las clases ‘CandyFrameLevelX’, ‘BoardPanelLevelX’ y ‘ScorePanelLevelX’ para cada uno de los niveles.

  En ScorePanelX se muestra la información relevante de cada nivel, para una mayor claridad del estado del juego (como por ejemplo la cantidad de movimientos restantes, las celdas que faltan pintar de dorado en GoldenBoard, etc).

  En cada BoardPanel se implementa el efecto visual al seleccionar un caramelo y el efecto dorado para el nivel 2. En el nivel 3 se indican los movimientos restantes para que exploten los caramelos bomba; y en el 4 se indica el tiempo que agrega cada caramelo al contador.

  En cuanto a las clases ‘CandyFrameLevelX’, cada nivel implementa los ScorePanel y BoardPanel correspondientes y maneja la interacción del usuario con el backend.

Backend

- En las clases Grid y CandyGame, se agregó el método “isValidMove()” dado que no era posible comprobar si un movimiento era válido o no sin ejecutarlo.

- En la clase GameState, se agregó la propiedad “maxMoves” junto con su getter y setter, además de los métodos “doOnMove()” y “getScorePanelData()”. Todo esto fue necesario para poder modularizar correctamente los states de cada nivel y para mantenerlos privados.

- En la clase CandyGame, se agregó el método “getState()” para obtener el state de cada juego y lograr una mejor modularización.

- En la carpeta level, se creó una clase abstracta “Level” para aprovechar al máximo su comportamiento a la hora de crear nuevos Levels, logrando así un mejor estilo.

- En la clase abstracta Element, se agregó la propiedad “number” que es usada en 2 de los 3 nuevos niveles implementados (junto con su getter y setter). Llegamos a la conclusión de que era lo mejor dado que el contenido de cada Cell es un Element en la implementación provista, y nos permite acceder a su propiedad de forma fácil y sin hacer casteos que puedan dar lugar a Exceptions.